SPRING BEAN LIFE CYCLE

http://www.tutorialspoint.com/spring/spring_bean_life_cycle.htm

The life cycle of a Spring bean is easy to understand. When a bean is instantiated, it may be required to perform some initialization to get it into a usable state. Similarly, when the bean is no longer required and is removed from the container, some cleanup may be required.

Though, there is lists of the activities that take place behind the scenes between the time of bean Instantiation and its destruction, but this chapter will discuss only two important bean lifecycle callback methods which are required at the time of bean initialization and its destruction.

To define setup and teardown for a bean, we simply declare the <bean> with **init-method** and/or **destroy-method** parameters. The init-method attribute specifies a method that is to be called on the bean immediately upon instantiation. Similarly, destroy-method specifies a method that is called just before a bean is removed from the container.

Initialization callbacks:

The *org.springframework.beans.factory.InitializingBean* interface specifies a single method:

```
void afterPropertiesSet() throws Exception;
```

So you can simply implement above interface and initialization work can be done inside afterPropertiesSet() method as follows:

```
public class ExampleBean implements InitializingBean {
   public void afterPropertiesSet() {
      // do some initialization work
   }
}
```

In the case of XML-based configuration metadata, you can use the **init-method** attribute to specify the name of the method that has a void no-argument signature. For example:

```
<bean
         />
```

Following is the class definition:

```
public class ExampleBean {
   public void init() {
      // do some initialization work
   }
}
```

Destruction callbacks

The *org.springframework.beans.factory.DisposableBean* interface specifies a single method:

```
void destroy() throws Exception;
```

So you can simply implement above interface and finalization work can be done inside destroy() method as follows:

```
public class ExampleBean implements DisposableBean {
   public void destroy() {
      // do some destruction work
   }
```

```
}
```

In the case of XML-based configuration metadata, you can use the **destroy-method** attribute to specify the name of the method that has a void no-argument signature. For example:

```
<bean
         />
```

Following is the class definition:

```
public class ExampleBean {
   public void destroy() {
      // do some destruction work
   }
}
```

If you are using Spring's IoC container in a non-web application environment; for example, in a rich client desktop environment; you register a shutdown hook with the JVM. Doing so ensures a graceful shutdown and calls the relevant destroy methods on your singleton beans so that all resources are released.

It is recommended that you do not use the InitializingBean or DisposableBean callbacks, because XML configuration gives much flexibility in terms of naming your method.

Example:

Let us have working Eclipse IDE in place and follow the following steps to create a Spring application:

Step	Description
1	Create a project with a name *SpringExample* and create a package *com.tutorialspoint* under the **src** folder in the created project.
2	Add required Spring libraries using *Add External JARs* option as explained in the *Spring Hello World Example* chapter.
3	Create Java classes *HelloWorld* and *MainApp* under the *com.tutorialspoint* package.
4	Create Beans configuration file *Beans.xml* under the **src** folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **HelloWorld.java** file:

```
package com.tutorialspoint;

public class HelloWorld {
   private String message;

   public void setMessage(String message){
      this.message  = message;
   }
   public void getMessage(){
      System.out.println("Your Message : " + message);
   }
   public void init(){
      System.out.println("Bean is going through init.");
   }
```

```
   public void destroy(){
      System.out.println("Bean will destroy now.");
   }
}
```

Following is the content of the **MainApp.java** file. Here you need to register a shutdown hook **registerShutdownHook()** method that is declared on the AbstractApplicationContext class. This will ensures a graceful shutdown and calls the relevant destroy methods.

```
package com.tutorialspoint;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
   public static void main(String[] args) {

      AbstractApplicationContext context =
                          new ClassPathXmlApplicationContext("Beans.xml");

      HelloWorld obj = (HelloWorld) context.getBean("helloWorld");
      obj.getMessage();
      context.registerShutdownHook();
   }
}
```

Following is the configuration file **Beans.xml** required for init and destroy methods:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <bean

       init-method="init" destroy-method="destroy">
       <property name="message" value="Hello World!"/>
   </bean>

</beans>
```

Once you are done with creating source and bean configuration files, let us run the application. If everything is fine with your application, this will print the following message:

```
Bean is going through init.
Your Message : Hello World!
Bean will destroy now.
```

Default initialization and destroy methods:

If you have too many beans having initialization and or destroy methods with the same name, you don't need to declare **init-method** and **destroy-method** on each individual bean. Instead framework provides the flexibility to configure such situation using **default-init-method** and **default-destroy-method** attributes on the <beans> element as follows:

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"
    default-init-method="init"
    default-destroy-method="destroy">

   <bean  >
       <!-- collaborators and configuration for this bean go here -->
```

```
    </bean>

</beans>
```